# **Change request log**

**Team**

Group 10:

Drew Smith

Nate Craver

**Change Request**

ID: fEMRFEMR-49 Password Constraints NeededFEMR-159

Description: When creating a new user, there is only one Password field. This leaves room for error if someone has a typo in their password. Add a second password field and make sure it matches the original before accepting the form.

## **Concept Location**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Ran fEMR in IntelliJ |  |
| 2 | We interacted with the system: after logging in we entered the users directory inside the admin directory | In order to get familiar with some of the features of the system, and identify the elements we had to change. |
| 3 | We searched for "password“ using the regular expression feature of the IDE's search tool. | To see what classes/functions were using the password as one of it’s variables |
| 4 | Most of the results came from the file CreateViewModel.java | Opened up the file to see the implementations |
| 5 | We inspected the class and the validate() function | We noticed the class was where we wanted to implement the mutator and accessor functions to add the functionality to our system |
| 5 | Once we knew where to implement our functions, we spectated other classes involved | To see how the functions are called to create a new user |
| 6 | We marked the class CreateViewModel as "located". | We confirmed this class had to be modified. |

**Time spent (in minutes):** 120

## **Impact Analysis**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We made a list of methods called by UsersController | The class calls CreateViewModel to create a new user |
| 2 | We inspected the data structures inside the class to see how the information is passed in | We realized that the UsersContorller class gets passed into a struct where the users information is held |
| 3 | We thought we needed to modify the UsersController class to call the class we modified | We realized that the UsersController class calls the whole class and not just an individual function so we did not have to make any changes to the class. But we did understand that our changes would in fact impact the system |

**Time spent (in minutes):** 60

## **Prefactoring**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Once we implemented out changes to the CreateViewModel class we noticed that the IDE suggested that the functions were never called | That is because no other classes individually used the newly created functions, only the whole class itself is called |
| 2 | After the previous change, we ran the system to see if our changes took effect | The system worked as planned and no pre-factoring steps were necessary |
| 3 | We committed our changes with git | Just in case we need to revert our changes. |

**Time spent (in minutes):** 10

## **Actualization**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We created a variable named confirmPassword within the CreateViewModel class | To store the users second password input |
| 2 | We then created an accessor/mutator functions | To be called from the class to get the users input |
| 3 | We edited the validate() function by adding an if statement | To make sure that the two passwords match in which the user inputs |
| 4 | We ran and tested the system | If the user enters in two different inputs, an error occurs |

**Time spent (in minutes):** 30

## **Postfactoring**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | When we implemented the accessor/mutator functions, it asked us to add a test method to a different class | Because the methods were never being used |
| 2 | After refactoring the test changes, we noticed that the refactoring was unnecessary | Because the functions did not need to called anywhere else. The whole class gets called to create a user |
| 3 | We committed and pushed our changes with git. | Just in case we need to revert our changes. |

**Time spent (in minutes):** 10

**Validation**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Test case defined:  Inputs:  Password: password1  Confirm Password: password1  Expected output:  New user has been created | This is the regular expected behavior.  The test passed. |
| 2 | Test case defined:  Inputs:  Password: password1  Confirm Password: password2  Expected output:  Error: passwords do not match | This is an exceptional behavior when the two passwords that the user inputs do not match  The test passed. |

**Time spent (in minutes):** 20

**Timing**

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 120 |
| Impact Analysis | 60 |
| Prefactoring | 10 |
| Actualization | 30 |
| Postfactoring | 10 |
| Verification | 20 |
| Total | 250 |

## 

## **Conclusions**

The most challenging part of this change request was to actually locate the class in which needed to be modified (Concept Location). After searching through numerous Java and Java Script files, we were finally able to locate where to implement the change. At first we were concerned that we would need to edit a Java Script file for the front-end of the change request but we soon realized that the code inside the Java files that we added would take care of that on its own. Since there was already a function for the user to input the password, it was pretty simple to create a new function in order for the user to confirm the password, as they were identical. Last, by implementing the added if-else function inside the validate function, everything worked fine. We did not really need to do any refactoring or post factoring techniques as they were unnecessary.